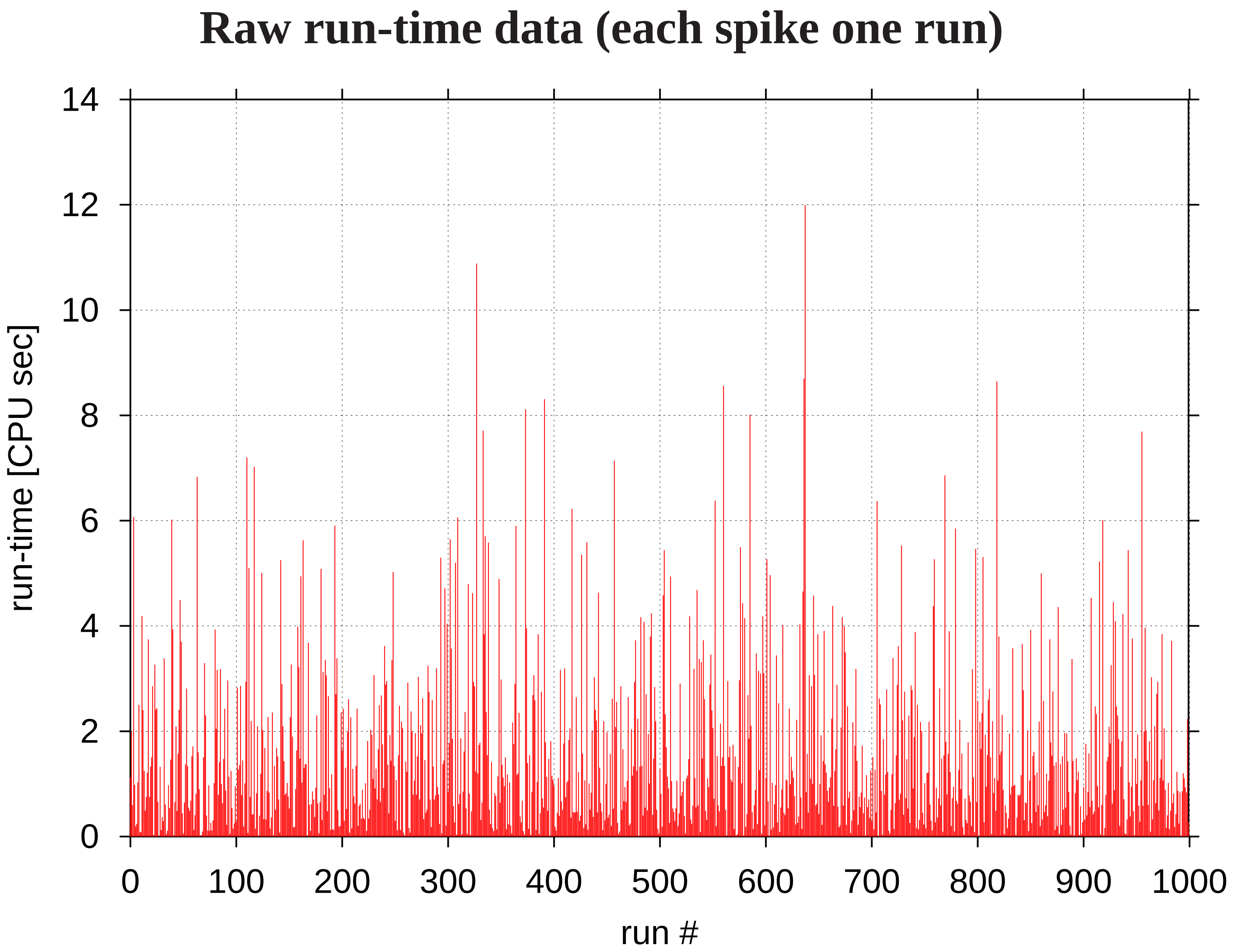
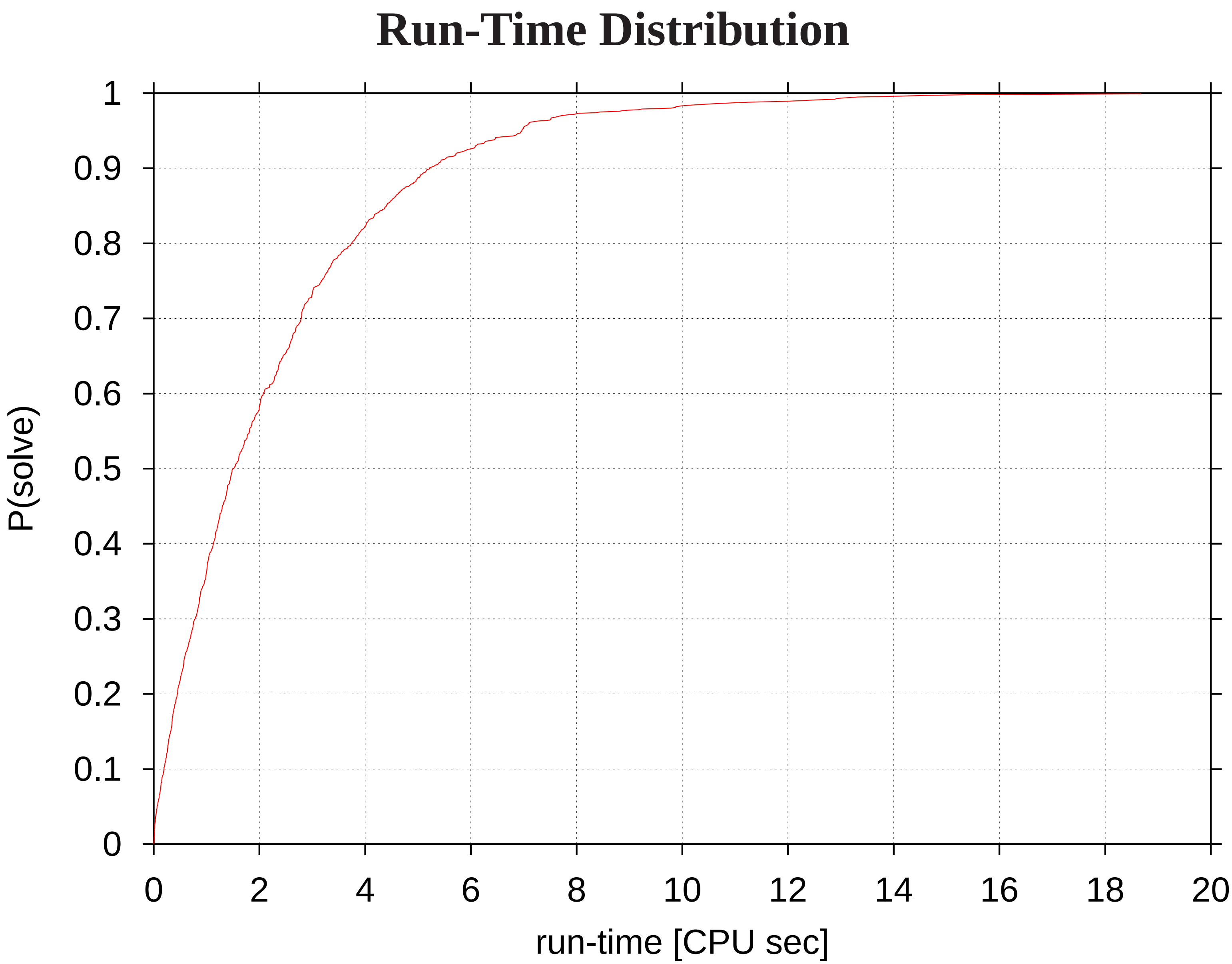
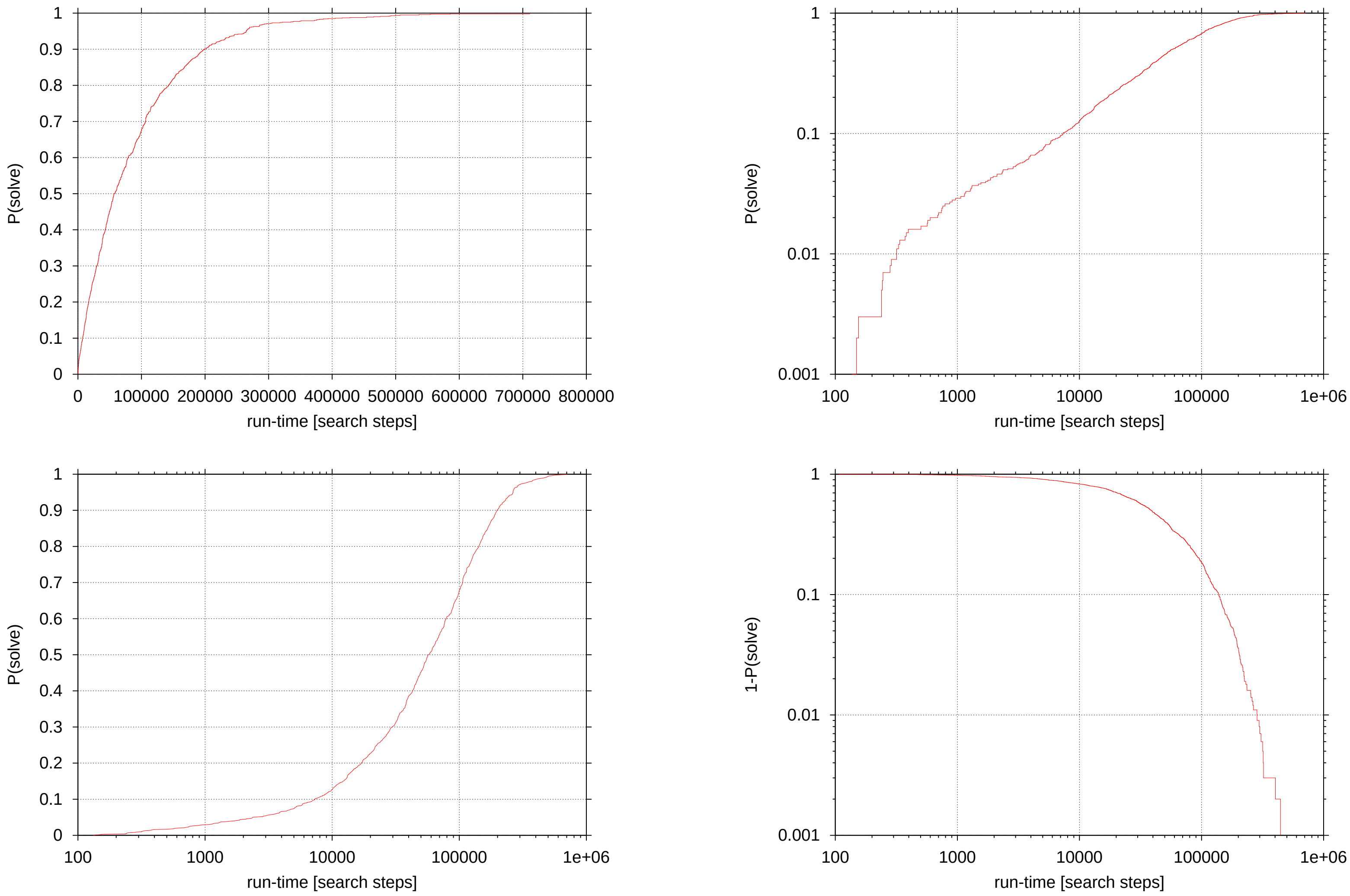


Raw run-time data (each spike one run)
run-time [CPU sec]
0
2
4
6
8
10
12
14
0
100
200
300
400
500
600
700
800
900
1000
run #



Run-Time Distribution
P(solve)
run-time [CPU sec]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16
18
20

RTD Graphs

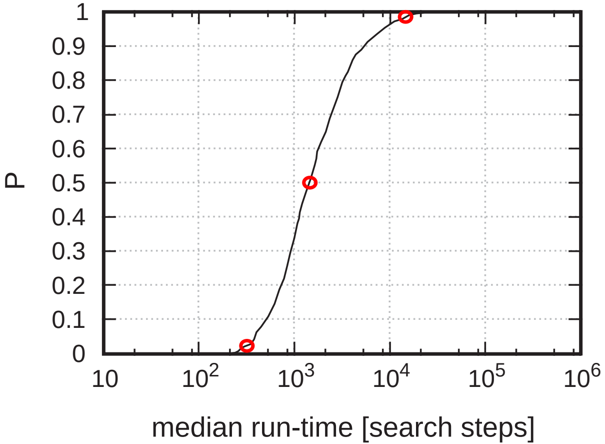


P(solve)
run-time [search steps]
P(solve)
run-time [search steps]
P(solve)
run-time [search steps]
1-P(solve)
run-time [search steps]

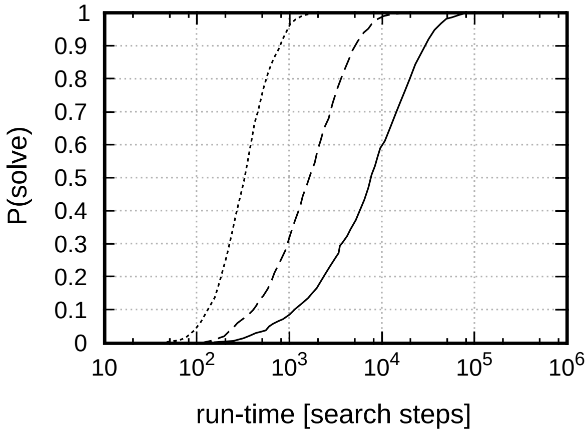
Protocol for obtaining the empirical RTD for an LVA A applied to a given instance π of a decision problem:

- Perform k independent runs of A on π with cutoff time t'. (For most purposes, k should be at least 50–100, and t' should be high enough to obtain at least a large fraction of successful runs.)
- Record number k' of successful runs, and for each run, record its run-time in a list L.
- Sort L according to increasing run-time; let $rt(j)$ denote the run-time from entry j of the sorted list $(j = 1, \ldots, k')$.
- Plot the graph $(rt(j), j/k)$, *i.e.*, the cumulative empirical RTD of A on π.

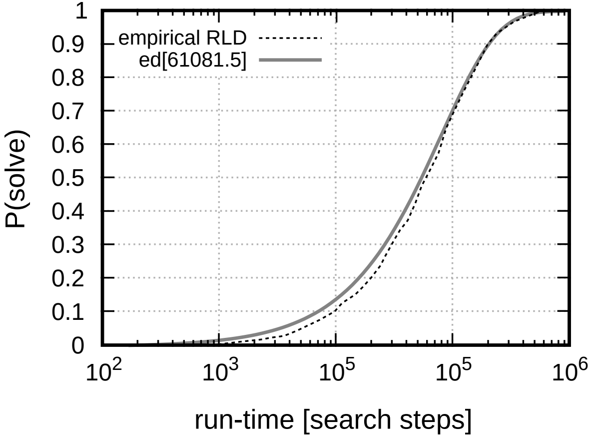
Distribution of median search cost for WalkSAT/SKC over set of 1000 randomly generated, hard 3-SAT instances:



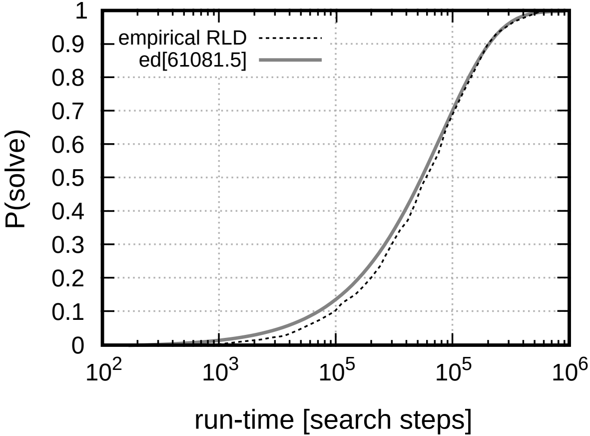
RTDs for WalkSAT/SKC, a prominent SLS algorithm for SAT, on three hard 3-SAT instances:



Approximation of an empirical RTD with an exponential distribution $ed[m](x) := 1 - 2^{-x/m}$:

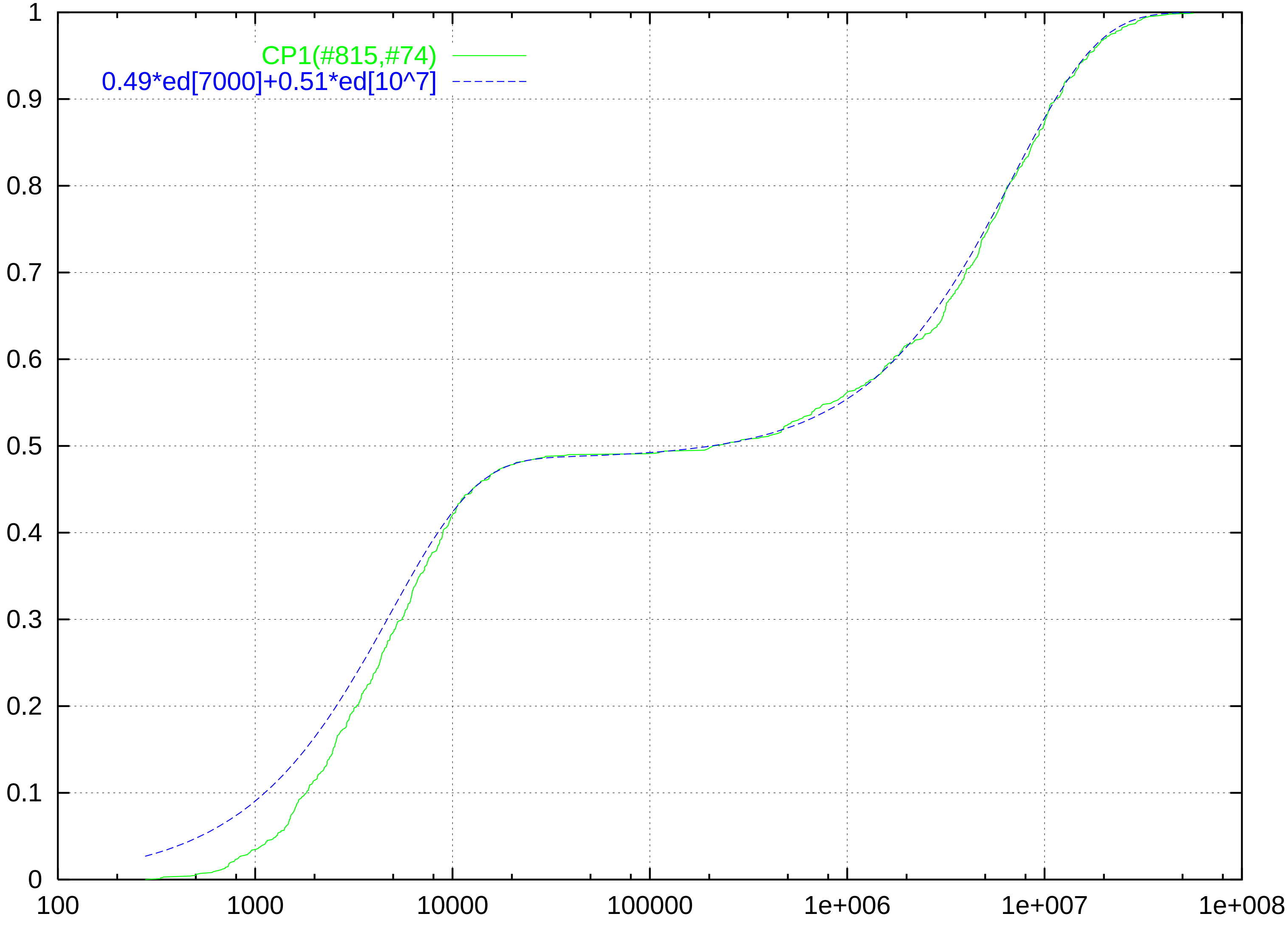


Approximation of an empirical RTD with an exponential distribution $ed[m](x) := 1 - 2^{-x/m}$:



The optimal fit exponential distribution obtained from the Marquardt-Levenberg algorithm passes the χ^2 goodness-of-fit test at $\alpha = 0.05$.

RTD Approximation with Mixture of Exponential Distributions

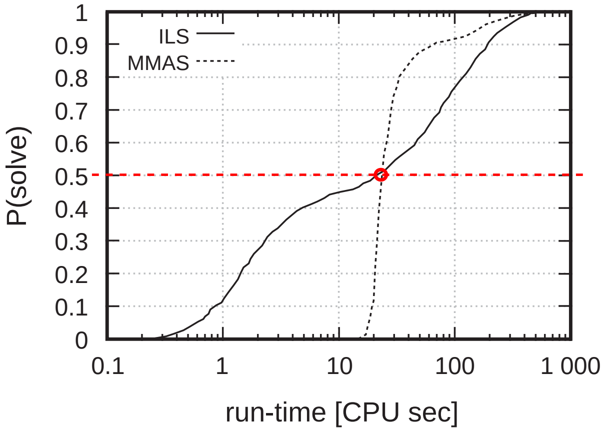


Performance differences detectable by the Mann-Whitney U-test for various sample sizes (sign. level 0.05, power 0.95):

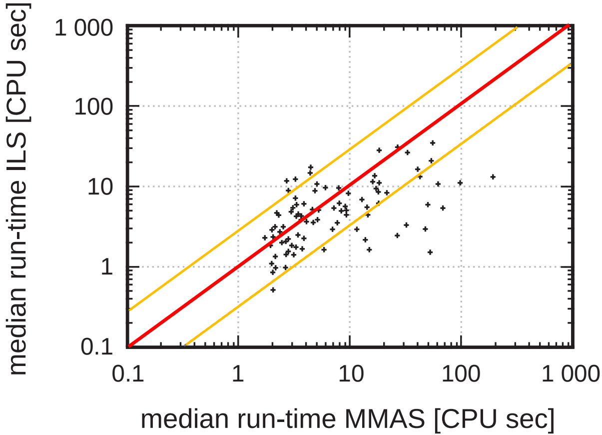
sample size	m_1/m_2
3 010	1.1
1 000	1.18
122	1.5
100	1.6
32	2
10	3

m_1/m_2 is the ratio between the medians of the two empirical distributions.

Example of crossing RTDs for two SLS algorithms for the TSP applied to a standard benchmark instance (1000 runs/RTD):



Correlation between median run-time for two SLS algorithms for the TSP over a set of 100 randomly generated instances:



10 runs per instance.

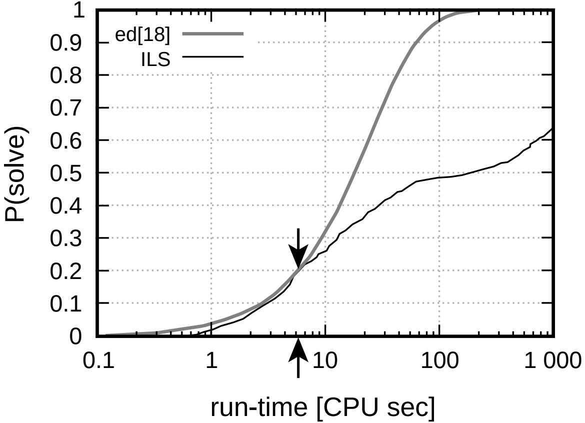
Performance improvements based on static restarts (1)

- Detailed RTD analyses can often suggest ways of improving the performance of a given SLS algorithm.

- *Static restarting*, *i.e.*, periodic re-initialisation after all integer multiples of a given cutoff-time t', is one of the simplest methods for overcoming stagnation behaviour.

- A static restart strategy is effective, *i.e.*, leads to increased solution probability for some run-time t'', if the RTD of the given algorithm and problem instance is less steep than an exponential distribution crossing the RTD at some time $t < t''$.

Example of an empirical RTD of an SLS algorithm on a problem instance for which static restarting is effective:



'ed[18]' is the CDF of an exponential distribution with median 18; the arrows mark the optimal cutoff-time for static restarting.

Performance improvements based on static restarts (2)

- To determine the optimal cutoff-time t_{opt} for static restarts, consider the left-most exponential distribution that touches the given empirical RTD and choose t_{opt} to be the smallest t value at which the two respective distribution curves meet.

  (For a formal derivation of t_{opt}, see page 193 of SLS:FA.)

- **Note:** This method for determining optimal cutoff-times only works *a posteriori*, given an empirical RTD.

- Optimal cutoff-times for static restarting typically vary considerably between problem instances; for optimisation algorithms, they also depend on the desired solution quality.

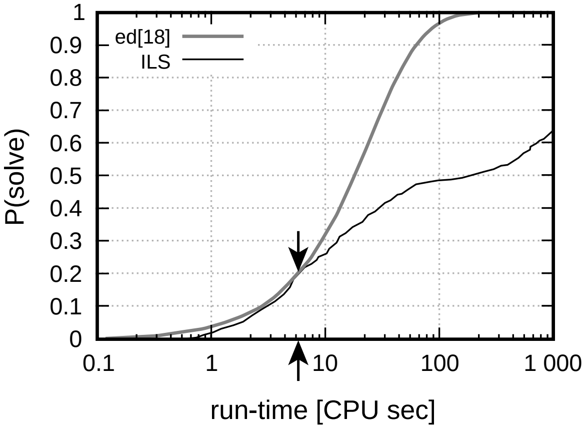
Performance improvements based on static restarts (1)

- Detailed RTD analyses can often suggest ways of improving the performance of a given SLS algorithm.

- *Static restarting*, *i.e.*, periodic re-initialisation after all integer multiples of a given cutoff-time t', is one of the simplest methods for overcoming stagnation behaviour.

- A static restart strategy is effective, *i.e.*, leads to increased solution probability for some run-time t'', if the RTD of the given algorithm and problem instance is less steep than an exponential distribution crossing the RTD at some time $t < t''$.

Example of an empirical RTD of an SLS algorithm on a problem instance for which static restarting is effective:



'ed[18]' is the CDF of an exponential distribution with median 18; the arrows mark the optimal cutoff-time for static restarting.

Performance improvements based on static restarts (2)

- To determine the optimal cutoff-time t_{opt} for static restarts, consider the left-most exponential distribution that touches the given empirical RTD and choose t_{opt} to be the smallest t value at which the two respective distribution curves meet.

  (For a formal derivation of t_{opt}, see page 193 of SLS:FA.)

- **Note:** This method for determining optimal cutoff-times only works *a posteriori*, given an empirical RTD.

- Optimal cutoff-times for static restarting typically vary considerably between problem instances; for optimisation algorithms, they also depend on the desired solution quality.

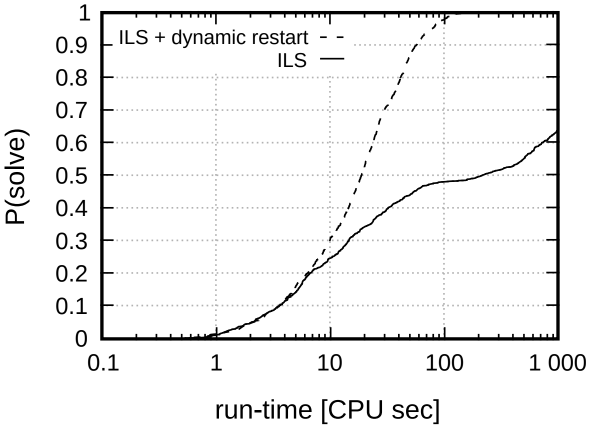
Overcoming stagnation using dynamic restarts

- *Dynamic restart strategies* are based on the idea of re-initialising the search process only when needed, *i.e.*, when stagnation occurs.

- *Simple dynamic restart strategy:* Re-initialise search when the time interval since the last improvement of the incumbent candidate solution exceeds a given threshold θ. (Incumbent candidate solutions are not carried over restarts.)

  θ is typically measured in search steps and may be chosen depending on properties of the given problem instance, in particular, instance size.

Example: Effect of simple dynamic restart strategy



Multiple independent runs parallelisation

- Any LVA A can be easily parallelised by performing multiple runs on the same problem instance π in parallel on p processors.

- The effectiveness of this approach depends on the RTD of A on π:

  Optimal parallelisation speedup of p is achieved for an exponential RTD.

- The RTDs of many high-performance SLS algorithms are well approximated by exponential distributions; however, deviations for short run-times (due to the effects of search initialisation) limit the maximal number of processors for which optimal speedup can be achieved in practice.

Speedup achieved by multiple independent runs parallelisation of a high-performance SLS algorithm for SAT: